Handwritten Character Recognition Based on Multi-Layer Perception

EEL5840 Final Project Report

1st Zhaoyang Chen (UFID: 9013-7235)
Department of Computer and Information Science and Engineering
University of Florida
Gainesville, FL 32611, USA
zhaoyang.chen@ufl.edu

***Abstract*—Design and Train a model with four layer neural network for handwritten character recognition and multi-class classification work with 90.13% accuracy on the training set and 66.24% accuracy on the validation set. Ten categories in total.**

***Index Terms*—Handwritten Character Recognition (HCR), Multi-Layer Perception (MLP), Multi-class Classification**

I. Introduction

As an important technical tool in the field of image recognition, Deep Learning has a broad application prospect, and the research on image recognition technology has important theoretical value and practical significance to promote the development of computer vision and artificial intelligence.

In this paper, my job is building the Multi-class Classifier based on 6720 handwritten character samples using Keras[1] and apply it to the HCR task. In-depth study and analyze the architecture and performance of Multi-Layer Perception from the practical and theoretical perspectives, respectively. The detailed work of this paper is as follows:

- Data preprocessing, including: data cleaning, morphological transformation, data standardization, encoding and scaling.
- Design and train the neural networks model.
- Model optimization and evaluation: fine tuning parameters, using dropout and data augmentation to avoid overfitting.

II. Implementation

A. Data preprocessing

Data preprocessing can be divided into the 5 steps. The raw data has 10 categories with a total of 6720 samples. The characters can be uppercase or lowercase. The labels are an array of integers, ranging from 0 to 9. The correspondence between label and class is shown in the Table 1.

Thanks to Dr. Catia S. Silva and TA. Haotian Yue for their help in the Machine Learning course this semester.

TABLE I
Label and Class

Label	0	1	2	3	4	5	6	7	8	9
Class	a	b	c	d	e	f	g	h	$	#

1) Data Cleaning: Check data with random sampling. Correct mislabeled images and remove unrecognizable images. During the scrutiny, the number of mislabeled samples be corrected is 54 and the number of unrecognizable samples been removed is 23.

2) Morphological Transformation: Apply the morphological techniques[2] on each samples, in order to remove the noise of the image and bring out the informative parts of the image. The sequential operations include: Inverting grayscale, Dilation, Opening and Closing. The randomly select 5 samples, before and after morphological transformation are compared as shown below (the first row is sample befor the transformation).

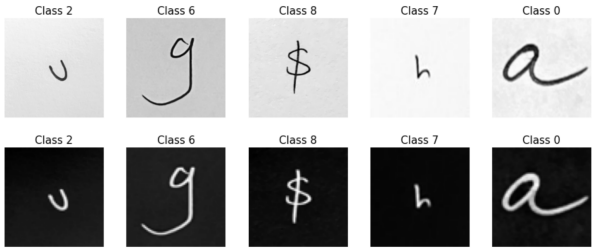

Fig. 1. Morphological transformation of characters

3) Normalization: Scale the pixel values to a range of 0 to 1. For each pixel value X, we have:

$$X_{normalized} = \frac{X_i - X_{\min}}{X_{\max} - X_{\min}}$$

4) Encoding: Use One-Hot Encoding to convert the categorical data to numerical data.

5) Resizing: The raw data are 300×300 images, stored in 2d numpy array. We need to convert them as 3d array and

resize to scale of 64×64. Reasonably resizing the data to a smaller size facilitates better tiling of the data into the neural network and prompts the training speed of the neural network without losing the features of the data.

B. Model Building

1) Network Layers: Construct the neural network by 4 layers.

- Input layer. In this layer, we need to input our dataset, consisting of 64×64 images. Flatten the images into one array Thus, we have 4096 nodes in this layer.
- Hidden layer 1. Reduce the number of nodes from 4096 to 512 nodes based on empirical observations.
- Hidden layer 2. Reduce the number of nodes from 512 to 64.
- Output layer. Reduce the number of nodes to 10, helping us evaluate the nodes against the label.

2) Activation Function: Choose the exponential linear unit[3] (ELU) as the activation function in hidden layers.

$$\mathrm{ELU}(x) = \begin{cases} x, & x > 0 \\ \alpha\left(\mathrm{e}^{x} - 1\right), & x \leqslant 0 \end{cases}$$

When the input data is positive, the ELU function is linear, its computation speed is faster than sigmoid and tanh. And during the backpropagation, if the input is negative, the gradient will not be exactly zero compared with RELU function.

3) Optimizer: For the optimizer, we decided to choose Adam as the optimizer. Adam[4] is a popular algorithm in the field of deep learning because it achieves good results fast (from the work of Diederik P. Kingma and Jimmy Lei Ba). Here is the plot about comparison of Adam to other optimization algorithms training the Multilayer Perceptron on training dataset in the first 100 epochs .

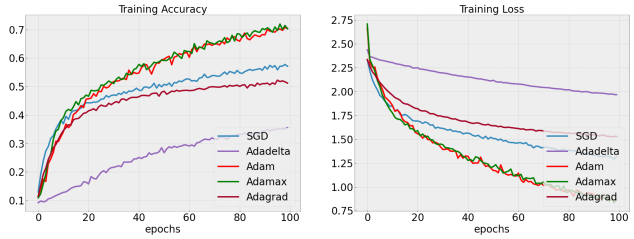

Fig. 2. Comparison of Adam with other optimizer on the training dataset

4) Loss Function: Binary cross-entropy is chosen as the loss function, since one-hot encoding is applied on the target labels, the integer labels has been converted to binary matrix. Here is the loss function.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\left(p\left(y_i\right)\right) + \left(1 - y_i\right) \cdot \log\left(1 - p\left(y_i\right)\right)$$

The function will compute that for each labels Y where $y_i \in Y$ it adds $log(p(y_i))$ to the loss if $y_i = 1$, conversely, it adds $log(1 - p(y))$ if $y_i = 0$. The model will compute score value for y_i and the predicted class is the y_i with the highest score.

III. Experiments

In the experiments part, two topic will be discussed. The first is the generalization ability of the model. The second part is the classification analysis.

A. Overfitting and Generalization

The difference in accuracy between training and validation accuracy is the sign of overfitting. In order to fight with overfitting, two ways are introduced: data augmentation and adding dropout to the model[5].

When apply dropout to a layer, it randomly drops out a number of output units from the layer during the training process. In my training process, we choose 8% output units dropout.

Due to limited training examples. Data augmentation works by generating additional training data from existing examples by augmenting them with random transformations, resulting in believable-looking images. This helps expose the model to more aspects of the data and generalize better. Considering the specificity of handwritten character, we only use randomly zoom to achieve data augmentation. Ways like flipping or rotating may cause the characters unrecognizable, and cropping will make the features of the data lost.

The following figure depicts the changes of accuracy and loss on training and validation dataset at the first 100 epochs. For the raw model (without adding dropout or data augmentation), it has poor generalization ability. Even though it reaches 0.8 accuracy on the training set, it behaves badly on the validation set. While the model with data augmentation and dropout shows better generalization ability.

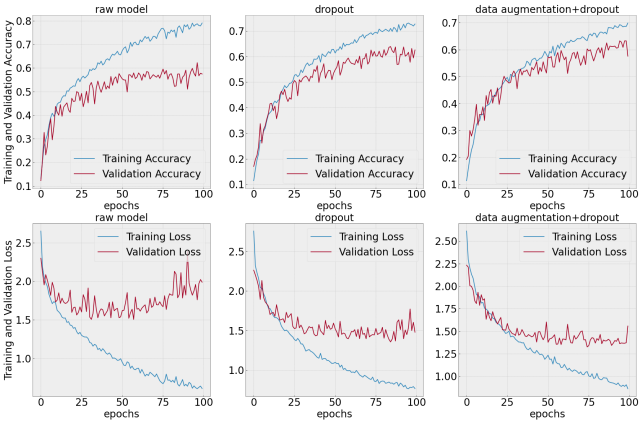

Fig. 3. Comparison of accuracy and loss on training and validation set (raw model, with dropout, with dropout and data augmentation)

B. Classification Analysis

For the total dataset (6697 samples in total, 23 samples are removed), stratified split the dataset by class to 90% training dataset with 6027 samples and 10% validation dataset with 670 samples. Apply both Dropout and data augmentation methods on the model, after 400 epochs, the accuracy on training dataset is around 90%, while the accuracy on validation dataset

TABLE II
CLASSIFICATION REPORT BY EACH CLASS

Class	Accuracy	Precision	recall	f1-score
A	0.66	0.66	0.66	0.66
B	0.57	0.56	0.57	0.57
C	0.72	0.51	0.72	0.59
D	0.64	0.84	0.64	0.72
E	0.54	0.85	0.54	0.66
F	0.74	0.65	0.74	0.70
G	0.63	0.57	0.63	0.60
H	0.63	0.67	0.63	0.65
$	0.67	0.67	0.67	0.67
#	0.63	0.66	0.63	0.64

is around 66%. And the classification report by each class is shown in table 2.

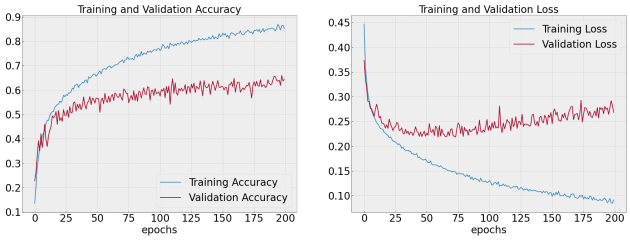

Fig. 4. Accuracy and loss on training and validation set (the first 200 epochs)

IV. CONCLUSION

In this paper, an MLP model is introduced and successfully classifies the handwritten characters. we applied the dropout and data augmentation methods to avoid overfitting. After fine tuning paramaters, the model reaches 66% on the validation dataset and 90% on training dataset. Due to the limited data resources, the difference between training and validation is still significant. Moreover, some noise is present in the original data. However we still verified that the dropout and data augmentation are powerful techniques to avoid overfitting. And the optimizer Adam is really efficient algorithm for gradient descent. Adam and Adamax give much higher performance than other optimizers.

REFERENCES

[1] Chollet F, others. Keras [Internet]. GitHub; 2015. Available from: https://github.com/fchollet/keras

[2] Bradski, G., 2000. The OpenCV Library. Dr. Dobb's Journal of Software Tools.

[3] Clevert, Djork-Arné et al. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)." arXiv: Learning (2016): n. pag.

[4] Borysenko O, Byshkin M. CoolMomentum: A method for stochastic optimization by Langevin dynamics with simulated annealing[J]. Scientific Reports, 2021, 11(1): 1-8.

[5] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.